AI for Text Analytics

Deep Learning and Universal Sentence-Embedding Models

1091AITA11
MBA, IMTKU (M2455) (8418) (Fall 2020)
Thu 3, 4 (10:10-12:00) (B206)

Min-Yuh Day

Associate Professor

Institute of Information Management, National Taipei University

https://web.ntpu.edu.tw/~myday
2020-12-24
<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
</table>
| 1 | 2020/09/17 | 人工智慧文本分析課程介紹  
(Course Orientation on Artificial Intelligence for Text Analytics) |
| 2 | 2020/09/24 | 文本分析的基礎：自然語言處理  
(Foundations of Text Analytics: Natural Language Processing; NLP) |
| 3 | 2020/10/01 | 中秋節 (Mid-Autumn Festival) 放假一天 (Day off) |
| 4 | 2020/10/08 | Python自然語言處理  
(Python for Natural Language Processing) |
| 5 | 2020/10/15 | 處理和理解文本  
(Processing and Understanding Text) |
| 6 | 2020/10/22 | 文本表達特徵工程  
(Feature Engineering for Text Representation) |
<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 2020/10/29</td>
<td>人工智慧文本分析個案研究</td>
<td>(Case Study on Artificial Intelligence for Text Analytics I)</td>
</tr>
<tr>
<td>8 2020/11/05</td>
<td>文本分類</td>
<td>(Text Classification)</td>
</tr>
<tr>
<td>9 2020/11/12</td>
<td>文本摘要和主題模型</td>
<td>(Text Summarization and Topic Models)</td>
</tr>
<tr>
<td>10 2020/11/19</td>
<td>期中報告</td>
<td>(Midterm Project Report)</td>
</tr>
<tr>
<td>11 2020/11/26</td>
<td>文本相似度和分群</td>
<td>(Text Similarity and Clustering)</td>
</tr>
<tr>
<td>12 2020/12/03</td>
<td>語意分析和命名實體識別</td>
<td>(Semantic Analysis and Named Entity Recognition; NER)</td>
</tr>
</tbody>
</table>
課程大綱 (Syllabus)

週次 (Week) 日期 (Date)  內容 (Subject/Topics)
13 2020/12/10 情感分析
   (Sentiment Analysis)
14 2020/12/17 人工智慧文本分析個案研究 II
   (Case Study on Artificial Intelligence for Text Analytics II)
15 2020/12/24 深度學習和通用句子嵌入模型
   (Deep Learning and Universal Sentence-Embedding Models)
16 2020/12/31 問答系統與對話系統
   (Question Answering and Dialogue Systems)
17 2021/01/07 期末報告 I (Final Project Presentation I)
18 2021/01/14 期末報告 II (Final Project Presentation II)
AI for Text Analytics

Text Mining “Knowledge Discovery in Textual Data”

Document Matching
Link Analysis
Search Engines

Information Retrieval

POS Tagging
Lemmatization
Word Disambiguation

Natural Language Processing

Web Mining
Web Content Mining
Web Structure Mining
Web Usage Mining

Data Mining
Classification
Clustering
Association

Statistics
Machine Learning
Management Science

Artificial Intelligence
Computer Science
Other Disciplines

Deep Learning and Universal Sentence-Embedding Models
Outline

• Universal Sentence Encoder (USE)

• Universal Sentence Encoder Multilingual (USEM)

• Semantic Similarity
NLP

Classical NLP

Deep Learning-based NLP

Documents → Language Detection → Pre-processing → Modeling → Output

Documents → Preprocessing → Dense Embeddings → Hidden Layers → Output Units → Output
Modern NLP Pipeline

Pre-processing

Documents

Language Detection

Pre-processed Documents

Tokenize

POS Tagging

Token Filter

EN

CN

Documents

Build Vocabulary

Pre-processed Documents

Bag-of-Words & Vectorization

Word Embeddings

(word2vec, doc2vec, GloVe)

Machine Learning

(Deep) Neural Network

Task / Output

Classification

Sentiment Analysis

Entity Extraction

Topic Modeling

Similarity

Modern NLP Pipeline

Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/
Deep Learning NLP

Documents → Preprocessing → Dense Word Embeddings → Deep Neural Network

Pre-generated Lookup OR Generated in 1st level of NeuralNet

Task / Output:
- Classification
- Sentiment Analysis
- Entity Extraction
- Topic Modeling
- Document Similarity

Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/
Natural Language Processing (NLP) and Text Mining

- Raw text
- Sentence Segmentation
- Tokenization
- Part-of-Speech (POS)
- Stop word removal
- Stemming / Lemmatization
- Dependency Parser
- String Metrics & Matching

Source: Nitin Hardeniya (2015), NLTK Essentials, Packt Publishing; Florian Leitner (2015), Text mining - from Bayes rule to dependency parsing
Data Science Python Stack

- TensorFlow
- Keras
- PyTorch
- Zipline
- DX Analytics
- PyAlgoTrade
- QuantLib
- PyTables
- NetworkX
- scikit-image
- StatsModels
- matplotlib
- seaborn
- pandas
- Bokeh
- plotly
- SymPy
- Scipy
- NumPy
- Python
- IPython
- Jupyter

Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#5
Universal Sentence Encoder (USE)

• The **Universal Sentence Encoder** encodes **text into high-dimensional vectors** that can be used for text classification, semantic similarity, clustering and other natural language tasks.

• The universal-sentence-encoder model is trained with a **deep averaging network (DAN)** encoder.

Source: [https://tfhub.dev/google/universal-sentence-encoder/4](https://tfhub.dev/google/universal-sentence-encoder/4)
Universal Sentence Encoder (USE) Semantic Similarity

"How old are you?" [0.3, 0.2, ...]
"What is your age?" [0.2, 0.1, ...]
"My phone is good." [0.9, 0.6, ...]

Source: https://tfhub.dev/google/universal-sentence-encoder/4
Universal Sentence Encoder (USE) Classification

"How old are you?"
"What is your age?"
"My phone is good."

Confidence is a question
(96%) "How old are you?"
(98%) "What is your age?"
(7%) "My phone is good."

Source: https://tfhub.dev/google/universal-sentence-encoder/4
Universal Sentence Encoder (USE)

```python
import tensorflow_hub as hub

embed = hub.Module("https://tfhub.dev/google/
   "universal-sentence-encoder/1")

embedding = embed(["The quick brown fox jumps over the lazy dog."])```
import tensorflow_hub as hub

module = hub.Module("https://tfhub.dev/google/universal-sentence-encoder-multilingual/1")

multilingual_embeddings = module(["Hola Mundo!", "Bonjour le monde!", "Ciao mondo!", "Hello World!", "Hallo Welt!", "Hallo Wereld!", "你好世界!", "Привет, мир!", "مرحبا بالعالم!"])
Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

Deep Learning and Universal Sentence-Embedding Models

Universal Sentence Encoder (USE)

- Source: Universal Sentence Encoder: https://tfhub.dev/google/universal-sentence-encoder/4

```python
[ ]
1 import tensorflow as tf
2 import tensorflow_hub as hub
3 import numpy as np
4 import pandas as pd
5 import os
6 import re
7 import matplotlib.pyplot as plt
8 import seaborn as sns
9
10 module_url = "https://tfhub.dev/google/universal-sentence-encoder/4"
11 #https://tfhub.dev/google/universal-sentence-encoder-large/5"
12 model = hub.load(module_url)
13 print ("module %s loaded" % module_url)
14 def embed(input):
15    return model(input)

[ ] module https://tfhub.dev/google/universal-sentence-encoder/4 loaded
```

[ ]
1 word = "Elephant"
2 sentence = "I am a sentence for which I would like to get its embedding."

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

https://tinyurl.com/aintpupython101
One-hot encoding

'The mouse ran up the clock' =

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>1</td>
<td>[0, 1, 0, 0, 0, 0, 0, 0],</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mouse</td>
<td>2</td>
<td>[0, 0, 1, 0, 0, 0, 0, 0],</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ran</td>
<td>3</td>
<td>[0, 0, 0, 1, 0, 0, 0, 0],</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>up</td>
<td>4</td>
<td>[0, 0, 0, 0, 1, 0, 0, 0],</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>1</td>
<td>[0, 1, 0, 0, 0, 0, 0, 0],</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock</td>
<td>5</td>
<td>[0, 0, 0, 0, 0, 1, 0, 0] ]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[0, 1, 2, 3, 4, 5, 6]
Word embeddings

Male-Female

Verb Tense

Country-Capital

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3
Word embeddings

The mouse ran up the clock

The mouse ran down

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3
Sequence to Sequence (Seq2Seq)

Source: https://google.github.io/seq2seq/
Transformer (Attention is All You Need)  
(Vaswani et al., 2017)

Transformer

Transformer
Encoder Decoder Stack

Transformer
Encoder Self-Attention

Transformer Encoder with Tensors

Word Embeddings

Transformer
Self-Attention Visualization

Transformer

Positional Encoding Vectors

Transformer
Self-Attention Softmax Output

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT

(Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

BERT input representation

BERT, OpenAI GPT, ELMo

Fine-tuning BERT on Different Tasks

(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

BERT Sequence-level tasks

(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

BERT Token-level tasks

(c) Question Answering Tasks:
SQuAD v1.1

(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Illustrated BERT

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

Semi-supervised Learning Step

Model: BERT
Dataset: Predict the masked word (language modeling)
Objective:

2 - Supervised training on a specific task with a labeled dataset.

Supervised Learning Step

Classifier

Model: BERT (pre-trained in step #1)

Dataset:

<table>
<thead>
<tr>
<th>Email message</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buy these pills</td>
<td>Spam</td>
</tr>
<tr>
<td>Win cash prizes</td>
<td>Spam</td>
</tr>
<tr>
<td>Dear Mr. Atreides, please find attached...</td>
<td>Not Spam</td>
</tr>
</tbody>
</table>

Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/
BERT Classification Input Output

Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/
BERT Encoder Input

Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/
BERT Classifier

Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/
Sentiment Analysis:
Single Sentence Classification

(b) Single Sentence Classification Tasks: SST-2, CoLA

"a visually stunning rumination on love"

Reviewer #1

That’s a **positive** thing to say

"reassembled from the cutting room floor of any given daytime soap"

Reviewer #2

That’s **negative**

<table>
<thead>
<tr>
<th>sentence</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>a stirring, funny and finally transporting re imagining of beauty and the beast and 1930s horror films</td>
<td>1</td>
</tr>
<tr>
<td>apparently reassembled from the cutting room floor of any given daytime soap</td>
<td>0</td>
</tr>
<tr>
<td>they presume their audience won't sit still for a sociology lesson</td>
<td>0</td>
</tr>
<tr>
<td>this is a visually stunning rumination on love, memory, history and the war between art and commerce</td>
<td>1</td>
</tr>
<tr>
<td>jonathan parker 's bartleby should have been the be all end all of the modern office anomie films</td>
<td>1</td>
</tr>
</tbody>
</table>

Movie Review Sentiment Classifier

“a visually stunning rumination on love” → Movie Review Sentiment Classifier → positive

Movie Review Sentiment Classifier

“a visually stunning rumination on love”

DistilBERT

Logistic Regression

positive

Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/
Movie Review Sentiment Classifier

Model Training

Movie Review Sentiment Classifier

DistilBERT

Already (pre-)trained

Logistic Regression

We will train in this tutorial

Step # 1 Use distilBERT to Generate Sentence Embeddings

Step #1: Use DistilBERT to embed all the sentences

Step #2: Test/Train Split for Model #2, Logistic Regression

Step #3 Train the logistic regression model using the training set

Step #3: Train the logistic regression model using the training set

Tokenization

[CLS] a visually stunning rum #имвination on love [SEP]

Tokenization

tokenizer.encode("a visually stunning rumination on love", add_special_tokens=True)

```
101  1037  17453  14726  19379  12758  2006  2293  102

[CLS]  a  visually  stunning  rumination  on  love  [SEP]
```

1) Break words into tokens

2) Add [CLS] and [SEP] tokens

3) Substitute tokens with their ids

"a visually stunning rumination on love"

Tokenization for BERT Model

Tokenization
DistilBertTokenizer

Input into the model

[CLS]  a  visually  stunning  rum  #ination  on  love  [SEP]

101  1037  17453  14726  19379  12758  2006  2293  102

3) substitute tokens with their ids
2) Add [CLS] and [SEP] tokens
1) Break words into tokens

Tokenize

“a visually stunning rumination on love”

Flowing Through DistilBERT (768 features)

Model Inputs: [CLS] 101 1037 17453 14726 19379 12758 2006 2293 102

Model Outputs: 

DistilBERT

Model #1 Output Class vector as Model #2 Input

Fine-tuning BERT on Single Sentence Classification Tasks

Model #1 Output Class vector as Model #2 Input

Model #2 Output

1 (positive)

0 (negative)

15%

85%

Model #2 Input

Model #1 Output

Logistic Regression

Logistic Regression Model to classify Class vector

```python
df = pd.read_csv('https://github.com/clairett/pytorch-sentiment-classification/raw/master/data/SST2/train.tsv',
delimiter='\t', header=None)
df.head()
```

<table>
<thead>
<tr>
<th></th>
<th>Text</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a stirring, funny and finally transporting re...</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>apparently reassembled from the cutting room f...</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>they presume their audience wo n't sit still f...</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>this is a visually stunning rumination on love...</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>jonathan parker 's bartleby should have been t...</td>
<td>1</td>
</tr>
</tbody>
</table>

Tokenization

tokenized = df[0].apply((lambda x: tokenizer.encode(x, add_special_tokens=True)))

# BERT Input Tensor

## BERT/DistilBERT Input Tensor

<table>
<thead>
<tr>
<th>Input sequences (reviews)</th>
<th>Tokens in each sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>101</td>
</tr>
<tr>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1,999</td>
<td>101</td>
</tr>
</tbody>
</table>

Processing with DistilBERT

```python
input_ids = torch.tensor(np.array(padded))
last_hidden_states = model(input_ids)
```
Unpacking the BERT output tensor

`last_hidden_states[0]`

BERT Output Tensor/predictions

66 Tokens in each sequence

2,000 Output rows (one per sequence)

768 Number of hidden units

Sentence to last_hidden_state[0]

```
<table>
<thead>
<tr>
<th>input_ids</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>1,999</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>last_hidden_states[0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

```
Batch
Tokenize all 2,000 sentences
Put each sentence in its own row
```

```
<table>
<thead>
<tr>
<th>101</th>
<th>1373</th>
<th>1745</th>
<th>14726</th>
<th>19370</th>
<th>12758</th>
<th>2006</th>
<th>2291</th>
<th>102</th>
<th>...</th>
<th>0</th>
</tr>
</thead>
</table>
```

```
“a visually stunning rumination on love”
```

BERT’s output for the [CLS] tokens

# Slice the output for the first position for all the sequences, take all hidden unit outputs
features = last_hidden_states[0][:,0,:].numpy()
The tensor sliced from BERT's output

Sentence Embeddings

Dataset for Logistic Regression (768 Features)

The features are the output vectors of BERT for the [CLS] token (position #0)

labels = df[1]
train_features, test_features, train_labels, test_labels = 
train_test_split(features, labels)
Score Benchmarks
Logistic Regression Model
on SST-2 Dataset

# Training
lr_clf = LogisticRegression()
lr_clf.fit(train_features, train_labels)

# Testing
lr_clf.score(test_features, test_labels)

# Accuracy: 81%
# Highest accuracy: 96.8%
# Fine-tuned DistilBERT: 90.7%
# Full size BERT model: 94.9%

Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/
<table>
<thead>
<tr>
<th>sentence</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>a stirring, funny and finally transporting re imagining of beauty and the beast and 1930s horror films</td>
<td>1</td>
</tr>
<tr>
<td>apparently reassembled from the cutting room floor of any given daytime soap</td>
<td>0</td>
</tr>
<tr>
<td>they presume their audience won't sit still for a sociology lesson</td>
<td>0</td>
</tr>
<tr>
<td>this is a visually stunning rumination on love, memory, history and the war between art and commerce</td>
<td>1</td>
</tr>
<tr>
<td>jonathan parker 's bartleby should have been the be all end all of the modern office anomie films</td>
<td>1</td>
</tr>
</tbody>
</table>

A Visual Notebook to Using BERT for the First Time

“a visually stunning rumination on love”

Reviewer #1

That’s a positive thing to say

“reassembled from the cutting room floor of any given daytime soap”

Reviewer #2

That’s negative

Pre-trained Language Model (PLM)

Semi-supervised Sequence Learning
context2Vec
Pre-trained seq2seq

ULMFiT
ELMo
Transformer
Bidirectional LM
Larger model
More data
Defense

GPT

Multi-lingual
MultiFiT
Transformer

Multi-task
XLM
UDify
MT-DNN

Cross-lingual
MASS
UniLM
Span prediction
Remove NSP
More data

Cross-modal
MT-DNNKD

+ Generation
Knowledge distillation
SpanBERT
RoBERTa

+Knowledge Graph
Permutation LM
Transformer-XL
More data

ERNE (Tsinghua)
ERNE (Baidu)

KnowBert
Neural entity linker

VideoBERT
CBT
ViLBERT
VisualBERT
B2T2
Unicoder-VL
LXMERT
VL-BERT
UNITER

By Xiaozhi Wang & Zhengyan Zhang @THUNLP

Source: https://github.com/thunlp/PLMpapers
Pre-trained Models (PTM)

PTMs

- Contextual?
  - Contextual
    - Non-Contextual
      - CBOW, Skip-Gram [129]
    - Contextual
      - ELMo [135], GPT [142], BERT [36]
    - LSTM
      - LM-LSTM [30], Shared LSTM [109], ELMo [135], CoVe [126]
    - Transformer Enc.
      - BERT [36], SpanBERT [117], XLNet [209], RoBERTa [117]
    - Transformer Dec.
      - GPT [142], GPT-2 [143]
    - Transformer
      - MASS [160], BART [100], XNLG [19], mBART [118]
  - Supervised
    - MT
      - CoVe [126]
    - LM
      - ELMo [135], GPT [142], GPT-2 [143], UniLM [39]
      - BERT [36], SpanBERT [117], RoBERTa [117], XLM-R [28]
    - MLM
      - TLM
      - XLM [27]
      - Seq2Seq MLM
      - MASS [160], T5 [144]
    - PLM
      - XLNet [209]
    - DAE
      - BART [100]
    - Unsupervised/Self-Supervised
      - RTD
      - CBOW-NS [129], ELECTRA [24]
      - CTL
      - NSP
      - BERT [36], UniLM [39]
      - SOP
      - ALBERT [93], StructBERT [193]

Pre-trained Models (PTM)

Transformers

State-of-the-art Natural Language Processing for TensorFlow 2.0 and PyTorch

- Transformers
  - pytorch-transformers
  - pytorch-pretrained-bert
- provides state-of-the-art general-purpose architectures
  - (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, CTRL...)
  - for Natural Language Understanding (NLU) and Natural Language Generation (NLG)
  - with over 32+ pretrained models
  - in 100+ languages
  - and deep interoperability between TensorFlow 2.0 and PyTorch.

Source: https://github.com/huggingface/transformers
## NLP Benchmark Datasets

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WMT 2014 EN-FR</td>
<td></td>
</tr>
<tr>
<td><strong>Text Summarization</strong></td>
<td>CNN/DM</td>
<td><a href="https://cs.nyu.edu/~kcho/DMQA/">https://cs.nyu.edu/~kcho/DMQA/</a></td>
</tr>
<tr>
<td></td>
<td>Newsroom</td>
<td><a href="https://summari.es/">https://summari.es/</a></td>
</tr>
<tr>
<td></td>
<td>Gigaword</td>
<td><a href="https://catalog.ldc.upenn.edu/LDC2012T21">https://catalog.ldc.upenn.edu/LDC2012T21</a></td>
</tr>
<tr>
<td><strong>Reading Comprehension</strong></td>
<td>ARC</td>
<td><a href="http://data.allenai.org/arc/">http://data.allenai.org/arc/</a></td>
</tr>
<tr>
<td></td>
<td>CliCR</td>
<td><a href="http://aclweb.org/anthology/N18-1140">http://aclweb.org/anthology/N18-1140</a></td>
</tr>
<tr>
<td></td>
<td>CNN/DM</td>
<td><a href="https://cs.nyu.edu/~kcho/DMQA/">https://cs.nyu.edu/~kcho/DMQA/</a></td>
</tr>
<tr>
<td></td>
<td>NewsQA</td>
<td><a href="https://datasets.maluuba.com/NewsQA">https://datasets.maluuba.com/NewsQA</a></td>
</tr>
<tr>
<td></td>
<td>RACE</td>
<td><a href="http://www.qizhexie.com/data/RACE_leaderboard">http://www.qizhexie.com/data/RACE_leaderboard</a></td>
</tr>
<tr>
<td></td>
<td>SQuAD</td>
<td><a href="https://rajpurkar.github.io/SQuAD_leaderboard">https://rajpurkar.github.io/SQuAD_leaderboard</a></td>
</tr>
<tr>
<td></td>
<td>NarrativeQA</td>
<td><a href="https://github.com/deepmind/narrativeqa">https://github.com/deepmind/narrativeqa</a></td>
</tr>
<tr>
<td></td>
<td>Quasar</td>
<td><a href="https://github.com/nyu-dl/Quasar">https://github.com/nyu-dl/Quasar</a></td>
</tr>
<tr>
<td></td>
<td>SearchQA</td>
<td><a href="https://github.com/nyu-dl/SearchQA">https://github.com/nyu-dl/SearchQA</a></td>
</tr>
<tr>
<td><strong>Question Answering</strong></td>
<td>AMR parsing</td>
<td><a href="https://amr.isi.edu/index.html">https://amr.isi.edu/index.html</a></td>
</tr>
<tr>
<td></td>
<td>ATIS (SQL Parsing)</td>
<td><a href="https://github.com/jkummerfeld/text2sql-data/tree/master/data">https://github.com/jkummerfeld/text2sql-data/tree/master/data</a></td>
</tr>
<tr>
<td></td>
<td>WikiSQL (SQL Parsing)</td>
<td><a href="https://github.com/salesforce/WikiSQL">https://github.com/salesforce/WikiSQL</a></td>
</tr>
<tr>
<td><strong>Semantic Parsing</strong></td>
<td>IMDB Reviews</td>
<td><a href="http://ai.stanford.edu/~amaas/data/sentiment/">http://ai.stanford.edu/~amaas/data/sentiment/</a></td>
</tr>
<tr>
<td></td>
<td>SST</td>
<td><a href="https://nlp.stanford.edu/sentiment/index.html">https://nlp.stanford.edu/sentiment/index.html</a></td>
</tr>
<tr>
<td></td>
<td>Yelp Reviews</td>
<td><a href="https://www.yelp.com/dataset/challenge">https://www.yelp.com/dataset/challenge</a></td>
</tr>
<tr>
<td></td>
<td>Subjectivity Dataset</td>
<td><a href="http://www.cs.cornell.edu/people/pabo/movie-review-data/">http://www.cs.cornell.edu/people/pabo/movie-review-data/</a></td>
</tr>
<tr>
<td><strong>Sentiment Analysis</strong></td>
<td>AG News</td>
<td><a href="http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html">http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html</a></td>
</tr>
<tr>
<td></td>
<td>DBpedia</td>
<td><a href="https://wiki.dbpedia.org/Datasets">https://wiki.dbpedia.org/Datasets</a></td>
</tr>
<tr>
<td></td>
<td>TREC</td>
<td><a href="https://trec.nist.gov/data.html">https://trec.nist.gov/data.html</a></td>
</tr>
<tr>
<td><strong>Text Classification</strong></td>
<td>SNLI Corpus</td>
<td><a href="https://nlp.stanford.edu/projects/snli/">https://nlp.stanford.edu/projects/snli/</a></td>
</tr>
<tr>
<td></td>
<td>MultiNLI</td>
<td><a href="https://www.nyu.edu/projects/bowman/multinli/">https://www.nyu.edu/projects/bowman/multinli/</a></td>
</tr>
<tr>
<td></td>
<td>SciTail</td>
<td><a href="http://data.allenai.org/scitail/">http://data.allenai.org/scitail/</a></td>
</tr>
<tr>
<td><strong>Natural Language Inference</strong></td>
<td>Proposition Bank</td>
<td><a href="http://propbank.github.io/">http://propbank.github.io/</a></td>
</tr>
<tr>
<td></td>
<td>OneNotes</td>
<td><a href="https://catalog.ldc.upenn.edu/LDC2013T19">https://catalog.ldc.upenn.edu/LDC2013T19</a></td>
</tr>
</tbody>
</table>

Summary

• Universal Sentence Encoder (USE)

• Universal Sentence Encoder Multilingual (USEM)

• Semantic Similarity
References


